Contents

Bioinformatics Lab Course - Report

Lab Course Algorithms in Bioinformatics

Dominik Drexler - 3526187

1 Motivation and Background

This lab report describes 5 of the probably most known algorithms used in bioinformatics. They challenge few of the most prominent problems in bioformatics. That are computation of alignments of two sequences, also called pairwise alignments. In this report the algorithms from Needleman-Wunsch and Gotoh are presented.
Another important problem is the computation of aligments of multiple sequences, called Multiple Sequence Alignment. It is computationally hard and therefore a heuristic approach called Feng-Dootlittle's Progressive Alignment is presented.
The problem of comparing ancestor-descendant relation is also computationally hard. Therefore so called Phylogenetic Trees (UPGMA, WPGMA) are computed using pairwise alignments.
The last problem is protein folding. Different foldings create different structures and result in different proteins. Finding such a stable structure is computationally hard as well. The Nussinov challenges this problem but is not used alot in practise.

2 The Needleman-Wunsch Algorithm

The Needleman-Wunsch Algorithm [6] is used for computing optimal pairwise alignments of two sequences with respect to a given scoring function. Pairwise alignments are used in biology to compare two indivials gene composition. Information about relation between gene functionality and an individual also open new ways to create personalized medicine that works best for the specific individual. The Needleman Wunsch algorithm is used alot because of the optimality and fast runtime.

2.1 Input of the algorithm

- A pair of sequences
- A scoring function, typically BLOSUM or PAM
- A gap score for aligning a symbol to an introduced gap

2.1.1 Relation to other algorithmic approaches

There is no better algorithm known, w.r.t. the runtime. The only improvement is a reduction in space complexity using the Hirschberg algorithm.
Another improvement in the biological sense is the Gotoh algorithm. This algorithm takes into account that it is biologically more likelier to have a lower amount of longer gaps compared to higher amount of shorter gaps in an alignment.
Another variation is slightly change the recurrence relation, such that the algorithm computes optimal local alignments.

2.2 Procedure

The Needleman Wunsch algorithm compute an optimal pairwise alignment from two given sequences. In order to describe the procedure of computing a pairwise alignment using Needleman Wunsch, it follows the definition of a pairwise alignment. A pairwise alignment is a two-row matrix where each row consists of one of the two input sequences. Additionally, to compare the quality of two alignments a score is attached. A definition for the scoring follows as well.

2.2.1 Definitions

First of all the definition of a pairwise alignment. This is what the algorithm return. It is a 2-row matrix where the rows correspond to the two given sequences and the columns correspond to the aligned symbols of the alignment.

Definition 1 (Pairwise Alignment) Let S_1, S_2 be two sequences. A pairwise alignment of S_1, S_2 is a 2-row matrix

$$\boldsymbol{A} = (A_{i,j}) \text{ with } i \in \{1, 2\} \text{ and } 1 \leq i \leq K$$

where i indicates the sequence and j is the column of the MSA with

- $\forall i, j : A_{i,j} \in \Sigma \cup \{-\}$
- $\forall i : (A_{i,1}, \ldots, A_{i,K})|_\Sigma = S_i$
- $\forall j : \neg(\forall i : A_{i,j} = -)$ □

To compare pairwise alignments, a scoring function assigns a score to an alignment. The score of an alignment is defined as follows:

Definition 2 (Score of a pairwise alignment) Let $\boldsymbol{A}$ be a pairwise alignment of sequences S_1, S_2 (rows) and K sites (columns).
Let $\mathcal{S} : \Sigma \times \Sigma \to \mathbb{R}$ be a scoring function given as a matrix.
The Pairwise Alignment Score of $\boldsymbol{A}$ is the sum of the score of all columns.

$$\mathcal{S}(\boldsymbol{A}) = \sum_{j=1}^{K} \mathcal{S}(\boldsymbol{A}_{1,j}, \boldsymbol{A}_{2,j})$$

□

Example 1

- Given Scoring Function $\mathcal{S}$ with

$$\mathcal{S}(x, y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{else} \end{cases}$$

- Given a pairwise alignment $\boldsymbol{A}$ with

$$\boldsymbol{A} = \begin{bmatrix} A & C & - \\ A & A & A \end{bmatrix}$$

- The score of $\boldsymbol{A}$ is computed as

$$\mathcal{S}(A) = \mathcal{S}(A, A + \mathcal{S}(C, A)) + \mathcal{S}(-, A) = 0 + 1 + 1 = 2$$

□

2.2.2 Procedure description

The algorithm uses dynamic programming. A matrix for the solutions of the subproblems is generated. For two given sequences x, y with length denoted by $|x|$ and $|y|$ the matrix has size $|x| + 1| \times |y| + 1$

- Recurrence relation:

$$D_{i,j} = \min \begin{cases} D_{i-1,j-1} + S(x_i, y_j) \\ D_{i-1,j} + \text{cost-gap-open} \\ D_{i,j-1} + \text{cost-gap-open} \end{cases}$$

$$D_{0,j} = j \cdot \text{cost-gap-open} \forall j \in \{0, \ldots, |x|\}$$

$$D_{i,0} = i \cdot \text{cost-gap-open} \forall i \in \{0, \ldots, |y|\}$$

Additionally a pointer to the cell used in the recursion is stored. The value in cell $D_{i,j}$ is the score of the optimal alignment for the subsequences $x_1, \ldots, x_i$ and $y_1, \ldots, y_j$. This means to compute a global optimal alignment the bottom right cell is the starting point for the traceback.
The traceback uses pointers to the cell used in the recursion. For a diagonal traceback arrow align the two corresponding characters. For a horizontal traceback arrow, align character from y to a gap in x. For a vertical traceback arrow, align character from x to gap in y

2.2.3 Theoretical analysis and Details

The solution returned by the algorithm is not unique, because multiple pairwise alignments may have the same score. The solution returned by the algorithm is optimal, w.r.t. the scoring function used.

2.2.4 Runtime analysis

The algorithms runtime is $O(nm)$ where n is the length of the first sequence and m is the length of the second sequence.
The algoriths space complexity is $O(nm)$.

2.3 Example

The following illustrates an example execution of the algorithm.

1. Needleman Wunsch example

   - Given Distance Scoring function S:

$$\begin{array}{c} \\ - \\ A \\ G \\ X \\ - \end{array} \begin{array}{c} \begin{array}{cccc} - & A & G & X & - \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & NIL \end{pmatrix} \end{array}$$

   - Given Sequences $S1 = AGCGCA$, $S2 = ACCA$

2. The dynamic programming matrix D

$$\begin{array}{c} \\ - \\ A \\ G \\ C \\ G \\ C \\ A \end{array} \begin{array}{c} \begin{array}{ccccc} - & A & C & C & A \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & -1 & 0 & 1 & 2 \\ 2 & 0 & -1 & 0 & 1 \\ 3 & 1 & -1 & -2 & -1 \\ 4 & 2 & 0 & -1 & -2 \\ 5 & 3 & 1 & -1 & -1 \\ 6 & 4 & 2 & 0 & -2 \end{bmatrix} \end{array}$$

3. global optimal alignment score $D_{4,3} = -2$ with pairwise alignment $\begin{bmatrix} A & G & C & G & C & A \\ A & - & C & - & C & A \end{bmatrix}$

3 The Gotoh Algorithm

The Gotoh Algorithm [5] is used for computing optimal pairwise alignments of two sequences with respect to a given scoring function. The scoring function uses a affine gap cost function which has a certain cost for opening a gap and for extending an existing gap. Pairwise alignments are used in biology to compare two indivials gene composition. Information about relation between gene functionality and an individual also open new ways to create personalized medicine that works best for the specific individuals. The Needleman Wunsch algorithm is used alot because of the optimality and fast runtime.

3.1 Input of the algorithm

- A pair of sequences
- A scoring function, typically BLOSUM or PAM
- A affine gap cost function $g(k) = \alpha + \beta \cdot k$

3.1.1 Relation to other algorithmic approaches

The Gotoh algorithm is very similar to the Needleman Wunsch algorithm with the improvement to return alignments that are biologically more likelier.
Another variation is to slightly change the recurrence relation, such that the algorithm computes optimal local alignments.

3.2 Procedure

The definitions of pairwise alignment and pairwise alignment score from Needleman Wunsch also apply here. Left is to describe the recurrence relation used to create the optimal pairwise alignment using a slightly changed traceback.
The Gotoh algorithm uses three Dynamic programming matrices of equal size to Needleman Wunsch.

- Matrix D: where $D_{i,j}$ is cost of the optimal alignment of subsequences $1 \dots x_i$ and $1 \dots y_j$ where x_i and y_j are aligned.
- Matrix P: where $P_{i,j}$ is the cost of the optimal alignment of subsequences $1 \dots x_i$ and $1 \dots y_j$ where x_i and a gap are aligned.

- Matrix Q: where $Q_{i,j}$ is the cost of the optimal alignment of subsequences $1 \dots x_i$ and $1 \dots y_j$ where y_j and a gap are aligned.

Example 2

- Given Scoring Function $\mathcal{S}$ with

$$\mathcal{S}(x, y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{else} \end{cases}$$

- Given a pairwise alignment $\boldsymbol{A}$ with

$$\boldsymbol{A} = \begin{bmatrix} A & C & - \\ A & A & A \end{bmatrix}$$

- The score of $\boldsymbol{A}$ is computed as

$$\mathcal{S}(A) = \mathcal{S}(A, A + \mathcal{S}(C, A)) + \mathcal{S}(-, A) = 0 + 1 + 1 = 2$$ □

3.2.1 Procedure description

The algorithm uses dynamic programming. A matrix for the solutions of the subproblems is generated. For two given sequences x, y with length denoted by $|x|$ and $|y|$ the matrix has size $|x| + 1| \times |y| + 1$

- Recurrence relation:

$$D_{i,j} = \min \begin{cases} D_{i-1,j-1} + S(x_i, y_j) \\ D_{i-1,j} + \text{cost-gap-open} \\ D_{i,j-1} + \text{cost-gap-open} \end{cases}$$

$$D_{0,j} = j \cdot \text{cost-gap-open} \forall j \in \{0, \dots, |x|\}$$

$$D_{i,0} = i \cdot \text{cost-gap-open} \forall i \in \{0, \dots, |y|\}$$

Additionally a pointer to the cell used in the recursion is stored. The value in cell $D_{i,j}$ is the score of the optimal alignment for the subsequences $x_1, \dots, x_i$ and $y_1, \dots, y_j$. This means to compute a global optimal alignment the bottom right cell is the starting point for the traceback.
The traceback uses pointers to the cell used in the recursion. For a diagonal traceback arrow align the two corresponding characters. For a horizontal traceback arrow, align character from y to a gap in x. For a vertical traceback arrow, align character from x to gap in y

3.2.2 Theoretical analysis and Details

The solution returned by the algorithm is not unique, because multiple pairwise alignments may have the same score. The solution returned by the algorithm is optimal, w.r.t. the scoring function used and affine gap cost function.

3.2.3 Runtime analysis

The algorithms runtime is $O(nm)$ where n is the length of the first sequence and m is the length of the second sequence.
The algoriths space complexity is $O(nm)$.

3.3 Example

The following illustrates an example execution of the algorithm.

1. Gotoh example

   - Given Distance Scoring function S:

     $$\begin{array}{c} \\ A \\ G \\ X \\ - \end{array} \begin{array}{c} \begin{array}{cccc} - & A & G & X \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & NIL \end{pmatrix} \end{array} \begin{array}{c} \\ - \\ 1 \\ 1 \\ 1 \\ NIL \end{array}$$

   - Affine gap cost function $g(k) = 1 + 1 \cdot k$
   - Given Sequences $S1 = AGCGCA$, $S2 = ACCA$

2. The dynamic programming matrix D

   $$\begin{array}{c} \\ A \\ G \\ C \\ G \\ C \\ A \end{array} \begin{array}{c} \begin{array}{ccccc} - & A & C & C & A \end{array} \\ \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 2 & -1 & 1 & 2 & 3 \\ 3 & 1 & -1 & 1 & 2 \\ 4 & 2 & 0 & -2 & 0 \\ 5 & 3 & 2 & 0 & -2 \\ 6 & 4 & 2 & 1 & 0 \\ 7 & 5 & 4 & 2 & 0 \end{bmatrix} \end{array}$$

   The dynamic programming matrix P

   $$\begin{array}{c} \\ A \\ G \\ C \\ G \\ C \\ A \end{array} \begin{array}{c} \begin{array}{ccccc} - & A & C & C & A \end{array} \\ \begin{bmatrix} 0 & inf & inf & inf & inf \\ 0 & 4 & 5 & 6 & 7 \\ 0 & 1 & 3 & 4 & 5 \\ 0 & 2 & 1 & 3 & 4 \\ 0 & 3 & 2 & 0 & 2 \\ 0 & 4 & 3 & 1 & 0 \\ 0 & 5 & 4 & 2 & 1 \end{bmatrix} \end{array}$$

   The dynamic programming matrix Q

   $$\begin{array}{c} \\ A \\ G \\ C \\ G \\ C \\ A \end{array} \begin{array}{c} \begin{array}{ccccc} - & A & C & C & A \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ inf & 4 & 1 & 2 & 3 \\ inf & 5 & 3 & 1 & 2 \\ inf & 6 & 4 & 2 & 0 \\ inf & 7 & 5 & 4 & 2 \\ inf & 8 & 6 & 4 & 3 \\ inf & 9 & 7 & 6 & 4 \end{bmatrix} \end{array}$$

3. global optimal alignment score $D_{4,3} = 0$ with pairwise alignment $\begin{bmatrix} A & G & C & G & C & A \\ A & - & - & C & C & A \end{bmatrix}$

This example shows the main difference to Needleman wunsch algorithm. The example in Needleman Wunsch has a slight difference with the gap function. Each gap has a constant cost and the length of a gapped part in the alignment is not taken into account. In this optimal alignment a gap of length two is prefered compared to perfectly matching the both C symbols.

4 The XPGMA Algorithm

This section describes how to generate guide trees such as Unweighted Pair Group Method with arithmetic Mean (UPGMA) or Weighted Pair Group Methid with arithmetic Mean (WPGMA) from a given set of sequences. They are used to describe the relation between a set of individuals. The result is used in other algorithms as guide. One famous example is the Feng-Doolittle algorithm for computing Multiple Sequence Alignments (MSA). The computation of a guide tree has polynomial time complexity.

4.1 Input of the algorithm

- A set of sequences
- A scoring function, typically BLOSUM or PAM
- A gap score for aligning a symbol to an introduced gap
- An algorithm to compute pairwise alignments (Typically Needleman-Wunsch)

4.2 Procedure

In the following the UPGMA algorithm is explained. By exchanging the cluster distance computation, one can derive the WPGMA algorithm. This formula will be given too.

4.2.1 Procedure description

Let $S = \{S_1, \ldots, S_N\}$ be a set of N sequences given as input to the algorithm.

1. Initialize set $\mathcal{C} = \{\{1\}, \ldots, \{N\}\}$
2. Initialize $dist(\{i\}, \{j\}) := dist(S_i, S_j)$ for all $1 \leq i, j \leq N$
3. Repeat $N - 1$ times
   (a) Find pair $(c, d) \in \mathcal{C} \times \mathcal{C}$ with $c \neq d$ where $dist(c, d)$ is minimal.
   $$d_{min} = dist(c, d)$$
   (b) Define a new cluster $e = c \cup d$
   $$\mathcal{C} = \mathcal{C} \setminus \{c, d\} \cup \{e\}$$
   (c) define node e with children c, d where e has distance $\frac{d_min}{2}$ to c, d

(d) define for all $f \in \mathcal{C}$ with $f \neq e$

$$dist(c \cup d, f) = dist(f, c \cup d) = \frac{dist(c, f) + dist(d, f)}{2}$$

Alternative distance for WPGMA:

$$dist(c \cup d, f) = dist(f, c \cup d) = \frac{|c| \cdot dist(c, f) + |d| \cdot dist(d, f)}{|c| + |d|}$$

4.2.2 Theoretical analysis and Details

The solution returned by the algorith is a good approximation to the biological relationship of the species given as a DNA sequence.

4.2.3 Runtime analysis

The algorithms runtime is $O(N^2)$ where N is the number of sequences.

4.3 Example

The following illustrates an example execution of the algorithm.

1. Initially $\mathcal{C} = \{a, b, c, d, e\}$
2. Precomputed distance matrix is:

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 0 & 3 & 12 & 12 & 9 \\ 3 & 0 & 13 & 13 & 10 \\ 12 & 13 & 0 & 6 & 7 \\ 12 & 13 & 6 & 0 & 7 \\ 9 & 10 & 7 & 7 & 0 \end{bmatrix} \end{array}$$

3. Iterate from $1, \ldots, 5 - 1$

   - Iteration 1:
     Select and merge a and $b \Rightarrow$ Newick $(a : 1.5, b : 1.5)$
     - $d(c, \{a, b\}) = \frac{d(a,c)+d(b,c)}{2} = \frac{12+13}{2} = 12.5$
     - $d(d, \{a, b\}) = \frac{12+13}{2}$
     - $d(e, \{a, b\}) = \frac{9+10}{2} = 9.5$

$$\begin{array}{c} \\ c \\ d \\ e \\ \{a, b\} \end{array} \begin{array}{c} \begin{array}{cccc} c & d & e & \{a, b\} \end{array} \\ \begin{bmatrix} 0 & 12.5 & 12.5 & 9.5 \\ 12.5 & 0 & 6 & 7 \\ 12.5 & 6 & 0 & 7 \\ 9.5 & 7 & 7 & 0 \end{bmatrix} \end{array}$$

   - Iteration 2:
     Select and merge c and $d \Rightarrow$ Newick $(c : 3, d : 3)$
     - $d(\{a, b\}, \{c, d\}) = \frac{d(\{a,b\},c)+d(\{a,b\},d)}{2} = \frac{12.5+12.5}{2} = 12.5$
     - $d(e, \{c, d\}) = \frac{7+7}{2} = 7$

$$\begin{array}{c} \\ e \\ \{a,b\} \\ \{c,d\} \end{array} \begin{array}{c} \begin{array}{ccc} e & \{a,b\} & \{c,d\} \end{array} \\ \begin{bmatrix} 0 & 12.5 & 9.5 \\ 12.5 & 0 & 7 \\ 9.5 & 7 & 0 \end{bmatrix} \end{array}$$

- Iteration 3:
  Select and merge $\{c,d\}$ and $e \Rightarrow$ Newick $((c:3,d:3):0.5,e:3.5)$
  - $d(\{a,b\},\{c,d,e\}) = \frac{d(\{a,b\},\{c,d\})+d(\{a,b\},e)}{2} = \frac{12.5+9.5}{2} = 11$

$$\begin{array}{c} \\ \{a,b\} \\ \{c,d,e\} \end{array} \begin{array}{c} \begin{array}{cc} \{a,b\} & \{c,d,e\} \end{array} \\ \begin{bmatrix} 0 & 11 \\ 11 & 0 \end{bmatrix} \end{array}$$

- Iteration 4:
  Select and merge $\{a,c\}$ and $\{c,d,e\} \Rightarrow$ Newick $((a:1.5,b:1.5):4,((c:3,d:3):0.5,e:3.5):2)$

Note: By exchanging the cluster distance compute to the WPGMA formula, one can generate WPGMAs

5 The Feng-Doolittle Algorithm

This chapter describes the Feng-Doolittle Algorithm [4] and the Sum-Of-Pairs score for Multiple Sequence Alignments [2]. The Feng-Dolittle Algorithms finds a not necessarily optimal solution for aligning multiple sequences at once. The result is called a Multiple Sequence Alignment and is used to create Phylogenetic Trees. These trees describe the ancestor-descendant relation between individuals of a species for example. But can also be used to describe relation between different species.
Difficulties with the underlying problem of computing Multiple Sequence Alignments is the computational complexity to create an optimal solution. This is why heurisitic approaches such as the Feng-Doolittle Algorithms were developed.

5.1 Input of the algorithm

- A collection of sequences.
- A guide tree, e.g. UPGMA or WPGMA
- A scoring function, either distance of similarity based function.

  In case of a similarity function, the pairwise alignment scores computed must be transformed to distance score again. The reason is that longer sequences should not be prefered just because of their length and the resulting higher possible similarity score.

5.1.1 Relation to other algorithmic approaches

- Compared to finding an optimal Multiple Sequence Alignment, which has runtime complexity of $O(n^N)$ (exponential) where n is the longest sequence and N is the amount of sequences, Feng-Doolittle has a relatively good (polynomial) runtime complexity.

5.2 Procedure

5.2.1 Definitions

First of all, the definition of a Multiple Sequence Alignment (MSA). This is what the algorithm returns. It is an alignment consisting of multiple sequences.

Definition 3 (Multiple Sequence Alignment (MSA)) Let $S_1, \ldots, S_N$ be N sequences. A multiple sequence alignment (MSA) of $S_1, \ldots, S_N$ is a matrix

$$\boldsymbol{A} = (A_{i,j}) \text{ with } 1 \leq i \leq N \text{ and } 1 \leq i \leq K$$

where i indicates the sequence and j is the column of the MSA with

- $\forall i, j : A_{i,j} \in \Sigma \cup \{-\}$
- $\forall i : (A_{i,1}, \ldots, A_{i,K})|_\Sigma = S_i$
- $\forall j : \neg(\forall i : A_{i,j} = -)$ □

Same as for pairwise alignment, it is necessary to assign a score to each multiple sequence alignment. The score makes it possible to order and compare multiple sequence alignments w.r.t to their quality.

Definition 4 (Sum-Of-Pairs Score SP [2]) Let $\boldsymbol{A}$ be a MSA with N sequences (rows) and K sites (columns).
Let $\mathcal{S}$ be a scoring function given as a matrix.
The Sum-Of-Pairs Score of $\boldsymbol{A}$ is the sum of the score of all columns.
A column c_j is scored by Sum-Of-Pairs (SP), denoted by $\mathcal{S}(c_j)$:

$$S(c_j) = \sum_{k<l} \mathcal{S}(A_{k,j}, A_{l,j})$$

□

Example 3

- Given Scoring Function $\mathcal{S}$ with

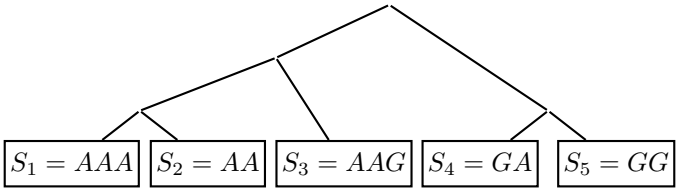
$$\mathcal{S}(x, y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{else} \end{cases}$$

- Given a MSA $\boldsymbol{A}$ with
  $\boldsymbol{A} = \begin{bmatrix} A & - & - \\ A & C & - \\ A & A & A \end{bmatrix}$
- The score of $\boldsymbol{A}$ is computed as
  $\mathcal{S}(\boldsymbol{A}) = \mathcal{S}\begin{pmatrix} \text{A} \\ \text{A} \\ \text{A} \end{pmatrix} + \mathcal{S}\begin{pmatrix} \text{-} \\ \text{C} \\ \text{A} \end{pmatrix} + \mathcal{S}\begin{pmatrix} \text{-} \\ \text{-} \\ \text{A} \end{pmatrix}$
- Where a column is scored as:
  $\mathcal{S}\begin{pmatrix} \text{-} \\ \text{C} \\ \text{A} \end{pmatrix} = \mathcal{S}\begin{pmatrix} \text{-} \\ \text{C} \end{pmatrix} + \mathcal{S}\begin{pmatrix} \text{-} \\ \text{A} \end{pmatrix} + \mathcal{S}\begin{pmatrix} \text{C} \\ \text{A} \end{pmatrix} = 1 + 1 + 1 = 3$ □

5.2.2 Procedure description

The following algorithmic description is taken from [3].

- Feng-Doolittle - Preprocessing/Input
  - – * When using similarity metric, convert it to distance metric!
    * Generate a guide tree, e.g. UPGMA (used in Feng&Doolittle)
    * Order of aligning sequences according to guide tree
  - – Example Guide Tree



- Feng-Doolittle - Algorithm
  - – Notation: S' denotes the sequence S where gaps are exchanged with extra Symbol X
  - – One of three possible operations may occur
    * Operation 1: Compute best pairwise alignment
      · Change occurences of gap symbol to X
    * Operation 2: Align a sequence S to an alignment A:
      · Compute pairwise alignment score between S and all sequences S'_i in A
      · Align S according to the sequence in the best pairwise alignment
      · Change occurences of gap symbol to X
    * Operation 3: Align an alignment A_1 to an alignment A_2
      · For each pair of sequences S'_1 in A_1 and S'_2 in A_2 compute pairwise alignment score
      · Align A_1 and A_2 according to the pairwise alignment with minimal distance.
      · Change occurences of gap symbol to X
- Guide tree traversal
  - – Guide tree consists of inner nodes, leaf nodes
  - – Traverse guide tree DFS
  - – Base case:
    * Leaf node: return the sequence
  - – Inductive case:
    * Case 1: Both children are leaf nodes
      · Apply operation 1 on the sequences returned by the leafs and return the alignment
    * Case 2: One Child is leaf and one node is inner node
      · Apply operation 2 on the sequence and the alignment and return the alignment

* Case 3: Both Children are inner nodes
  - Apply operation 3 on the alignments and return the alignment

After applying the last operator at the root node, change back the X symbol to a gap symbol and return the Multiple Sequence Alignment.

5.2.3 Remark on the special character X

The character X is used as a special neutral element where the score for aligning X with any other letter (including gap) is 0. This leads to making a gap more likelier to align to an already existing gap, represented as an X. In Feng-Doolittles paper, he called it, Once a gap always a gap!. As an example why this happens see the following scoring function and two alignment with their repective scorings. The next example shows this.

Example 4

- The distance scoring function, given as a matrix is:

$$\begin{array}{c} \\ A \\ G \\ X \\ - \end{array} \begin{array}{c} \begin{array}{cccc} - & A & G & X \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & NIL \end{pmatrix} \end{array}$$

- Alignments of the sequence $S_2 = AA$ to a sequence $S_1' = AXG$ with their respective scores:

$$S\begin{pmatrix} \text{A} & \text{X} & \text{G} \\ \text{A} & \text{-} & \text{A} \end{pmatrix} = -1 + 0 + 0 = -1$$

$$S\begin{pmatrix} \text{A} & \text{X} & \text{G} \\ \text{A} & \text{A} & \text{-} \end{pmatrix} = -1 + 0 + 1 = 0$$

- One can see that X aligned with a gap leads to better (distance) score □

5.2.4 Theoretical analysis and Details

The solution returned by the algorithm is not unique, because multiple pairwise alignments may have the same score. The solution returned by the algorithm is not the optimal solution, w.r.t. the Sum-Of-Pairs score.

5.2.5 Runtime analysis

The algorithm worst case runtime occurs in the case of a perfectly balanced tree.

$$\begin{aligned} T(N) &= 2 \cdot T(\frac{N}{2}) + \underbrace{\frac{N^2}{4}}_{\#\text{pairwise seq.}} \cdot \underbrace{K^2}_{\text{cost for pairwise align}} \\ &\Rightarrow T(N) = \Theta(N^2 K^2) \end{aligned}$$

5.3 Example

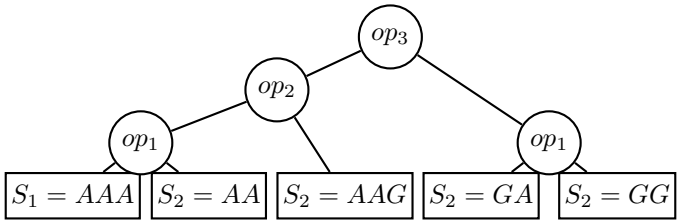
The following illustrates an example execution of the algorithm for a collection of five sequences.

1. Progressive alignment example

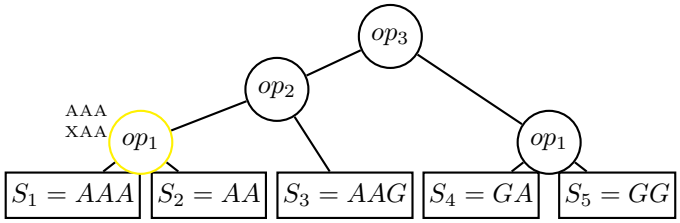
   - Given Distance Scoring function S:

$$\begin{array}{c} \\ A \\ G \\ X \\ - \end{array} \begin{array}{c} \begin{array}{cccc} - & A & G & X & - \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & NIL \end{pmatrix} \end{array}$$

   - Given Sequences $S1 = AAA$, $S2 = AA$, $S3 = AAG$, $S4 = GG$, $S5 = GA$
   - Guide tree as before:

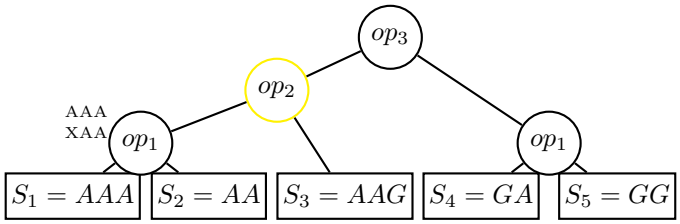


2. Progressive alignment example - applying operator 1



   - Operation 1: Compute best pairwise alignment
     - Change occurences of gap symbol to X
   - $\Rightarrow group_1 = \begin{bmatrix} A & A & A \\ X & A & A \end{bmatrix} \begin{matrix} S_1' \\ S_2' \end{matrix}$
   - Not necessarily unique!

3. Progressive alignment example - applying operator 2



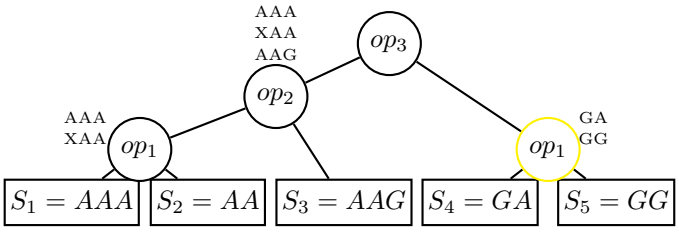
   - Operation 2: Align a sequence S_1 to an alignment A:
     - Recompute pairwise alignment score between S_1 and all sequences S_2' in A
     - Align S according to the sequence in the best pairwise alignment
     - Change occurences of gap symbol to X

- $group_2 = \begin{bmatrix} A & A & A \\ A & A & G \end{bmatrix} \begin{matrix} S'_1 \\ S_3 \end{matrix}$ with $\mathcal{S}(group_2) = -1$ (better)
- $group_3 = \begin{bmatrix} X & A & A & - \\ - & A & A & G \end{bmatrix} \begin{matrix} S'_2 \\ S_3 \end{matrix}$ with $\mathcal{S}(group_2) = 0$

4. Progressive alignment example - combining
   - $group_1 = \begin{bmatrix} A & A & A \\ X & A & A \end{bmatrix} \begin{matrix} S'_1 \\ S'_2 \end{matrix}$
   - $group_2 = \begin{bmatrix} A & A & A \\ A & A & G \end{bmatrix} \begin{matrix} S'_1 \\ S_3 \end{matrix}$
   - Align S_3 to $group_1$ through alignment $group_2$

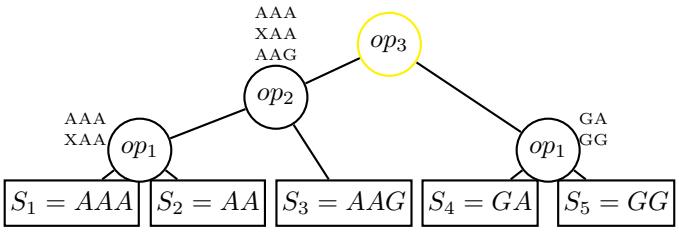
     $group_4 = \begin{bmatrix} A & A & A \\ X & A & A \\ A & A & G \end{bmatrix} \begin{matrix} S'_1 \\ S'_2 \\ S'_3 \end{matrix}$

5. Progressive alignment example - applying operator 1



   - Operation 1: Compute best pairwise alignment
     - Change occurences of gap symbol to X
   - $\Rightarrow group_5 = \begin{bmatrix} G & G \\ G & A \end{bmatrix} \begin{matrix} S'_4 \\ S'_5 \end{matrix}$

6. Progressive alignment example - applying operator 3



   - Operation 3: Align an alignment A_1 to an alignment A_2
     - For each pair of sequences S'_1 in A_1 and S'_2 in A_2 compute pairwise alignment score
     - Align A_1 and A_2 according to the pairwise alignment with minimal distance.

7. Progressive alignment example - applying operator 3 (2)

- $group_4 = \begin{bmatrix} A & A & A \\ X & A & A \\ A & A & G \end{bmatrix} \begin{matrix} S'_1 \\ S'_2 \\ S'_3 \end{matrix}$
- $group_5 = \begin{bmatrix} G & G \\ G & A \end{bmatrix} \begin{matrix} S'_4 \\ S'_5 \end{matrix}$
- For all pairs $(Q_1, Q_2) \in \{S'_1, S'_2, S'_3\} \times \{S'_4, S'_5\}$ the alignment with best score must not be unique
- Again: introducing a new gap costs us something (column 1)
- Therefore one possible best alignment is $group_6 = \begin{bmatrix} X & A & A \\ X & G & A \end{bmatrix} \begin{matrix} S'_2 \\ S'_5 \end{matrix}$

8. Progressive alignment example - combining
   - $group_4 = \begin{bmatrix} A & A & A \\ X & A & A \\ A & A & G \end{bmatrix} \begin{matrix} S'_1 \\ S'_2 \\ S'_3 \end{matrix}$
   - $group_5 = \begin{bmatrix} G & G \\ G & A \end{bmatrix} \begin{matrix} S'_4 \\ S'_5 \end{matrix}$
   - $group_6 = \begin{bmatrix} X & A & A \\ X & G & A \end{bmatrix} \begin{matrix} S'_2 \\ S'_5 \end{matrix}$
   - Align $group_4$ to $group_5$ through alignment $group_6$
   - $group_7 = \begin{bmatrix} A & A & A \\ X & A & A \\ A & A & G \\ X & G & G \\ X & G & A \end{bmatrix} \begin{matrix} S'_1 \\ S'_2 \\ S'_3 \\ S'_4 \\ S'_5 \end{matrix}$
9. Progressive alignment example - result
   - Exchange back X with $-$
   - The resulting MSA is:

     $group_7 = \begin{bmatrix} A & A & A \\ - & A & A \\ A & A & G \\ - & G & G \\ - & G & A \end{bmatrix} \begin{matrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \end{matrix}$

6 The Nussinov Algorithm

6.1 Input of the algorithm

- A RNA sequence

6.2 Procedure

The Nussinov Algorithm [7] computes an optimal secondary structure with as much base pairs as possible

6.2.1 Procedure description

The algorithm uses dynamic programming. A matrix for the solutions of the subproblems is generated. For the given sequence x the length is denoted by $|x|$. The Dynamic Programming matrix D has a shape of $|x| + 1 \times |x| + 1$

- Recurrence relation:

$$\forall i, j \text{ with } i > 0 : D_{i,j} = \max \begin{cases} D_{i,j-1} \\ \max_{1 \leq k < (k-1)} D_{i,k-1} + D_{k+1,j-1} + \sigma(x_k, x_j) \end{cases}$$

where

$$\sigma(x_k, x_j) = \begin{cases} 1 & \text{if } x_k \text{ and } x_j \text{ are complementary base pairs} \\ 0 & \text{otherwise} \end{cases}$$

$$\forall i, j \text{ with } i + 1 >= j : D_{i,j} = 0$$

Additionally a pointer to the cell(s) used in the recursion is stored.

6.2.2 Theoretical analysis and Details

The solution returned by the algorithm is not unique, because multiple tracebacks cases may have the same optimal value. The solution returned by the algorithm is optimal, w.r.t. the maximal number of base pairs.

6.2.3 Runtime analysis

The algorithms runtime is $O(N^3)$ where N is the length of the given sequence.
The algorithms space complexity is $O(N^2)$.

6.3 Example

The following illustrates an example execution of the algorithm.

1. Nussinov example [1]

   - Given Sequence $S = GGUCCAC$

2. The dynamic programming matrix D

	$-$	G	G	U	C	C	A	C
G	0	0	0	1	1	2	2	2
G		0	0	0	1	1	1	2
U			0	0	0	0	1	1
C				0	0	0	0	0
C					0	0	0	0
A						0	0	0
C							0	0

3. Optimal structures:

   - .((..))
   - (.(..))
   - ((..).)
   - ((.)..)
   - ((.))..

   where each pair of ( ) at the same depth indicates base pairing at those positions in the given sequence.

7 Summary

In this report a collection of five algorithms was explained.
For computing pairwise alignments, the Needleman-Wunsch and the Gotoh algorithm was explained. They are both polynomial time and space algorithms. The main difference is that gotoh uses an affine gap function to favor lower amount of longer gaps compared to higher amount of shorter gaps.
The third algorithm is the construction of phylogenetic trees (UPGMA, WPGMA) which represent the relationship between individuals. It is using the previous mentioned pairwise alignment algorithms and runs in polynomial time.
Another alignment problem is the computation of Multiple Sequence Alignments. Finding an optimal solution is possible in exponential time. This is infeasible for large dataset. As a result heuristic approaches such as the Feng-Doolittle algorithm for progressive alignment were found. It uses a phylogenetic tree to guide the algorithm. The resulting running time is polynomial.
The last algorithm is the Nussinov algorithm. It computes the maximal number of possible base pairings in an RNA sequence. The biological meaning is relatively low but the algorithm is a baseline for other algorithms that tackle the problem of computing protein foldings.

References

[1] Rna tools uni freiburg. `http://rna.informatik.uni-freiburg.de/`.

[2] Humberto Carillo and David Lipman. The multiple sequence alignment problem in biology.

[3] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. Biological sequence analysis - probabilistic models of protein and nucleic acids.

[4] Da-Fei Feng and Russell F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees.

[5] Osamu Gotoh. An improved algorithm for matching biological sequences, 1982.

[6] Saul Needleman and Christian Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins, 1970.

[7] Ruth Nussinov, George Pieczenik, Jerrold Griggs, and Daniel Kleitman. Algorithms for loop matchings.